Сделал Терентиев Антон студент группы СКБ 223

Отчёт

Приложение представляет собой консольное приложение-калькулятор, которое принимает на вход флаг, по которому определяется, как она будет работать. При вводе флага –forward пользователю необходимо будет ввести выражение. Оно будет высчитываться через функцию forward. Основой для этого решения сал стек (так как нам нельзя было использовать данный класс, пришлось создавать собственные два класса – стек float-ов и стек char-ов. Соответственно для каждого класса я прописал необходимые функции: add – добавление элемента, top – возвращает крайний элемент, pop – удаляет крайний элемент и size – возвращает количество элементов (подробнее про работу на защите отвечу, если она будет, но здесь всё просто). Далее реализовал несколько методов типа bool, чтобы не загромождать if-ы условиями. Потом я удаляю все пробелы из записи, чтобы не мешали. После чего я прохожусь по всей сроке задачи посимвольно. Цикл for я бы лучше объяснил вживую, но раз надо, то дам хоть какое-то описание здесь. У нас стоит иф, который разделяет символы на цифры и не цифры. Если цифра или –(отрицательное число), то записывается в строковую переменную tmpn. Если знак, то запускается элс. Первым делом проверяем, записано ли что-нибудь в строковую переменную tmpn. Если да, то закидываем в числовой стек, иначе ничего. После этого проверяем, пуст ли стек символов/действий, если да, то заносим туда знак , иначе проверяем является ли символ открывающей скобку ‘(‘ . Эта проверка необходима, чтобы не проверять всё остальное, а просто занести скобку в стек символов/операций. Потом проверяем на закрывающую скобку. Если это выполняется, то мы заходим в цикл while и выполняем все операции задом наперёд. Подробнее объясню вживую, но я беру крайний элемент стека чисел, записываю в tmp2, удаляю его, беру следующий крайний элемент, записываю его в tmp1 и опять удаляю. Потом получаю символ из стека действий и добавляю в стек чисел результат выражения tmp1 -действие- tmp2. Почему так, потому что мы получаем элементы задом наперёд, а это важно учитывать при вычитании и делении. С остальными действиями аналогично. Выполняю я это пока крайний элемент в стеке действий не будет открывающей скобкой. В таком случае я выхожу из while и удаляю из стека открывающую скобку и закидываю знак, который всё это время хранился в char c. Если всё таки не скобка, то проверяем приоритет крайней операции в стеке через мой метод. Если это \* или /, то метод вернёт true. Тогда выполняем умножение или деление и закидываем обратно в стек результат аналогично тому, как я описал выше. Если же метод приоритета вернул false, то проверяем на приоритет текущее действие в char c. Если приоритет “низкий”, то через цикл while выполняем все действия сложения/вычитания, пока крайней не будет приоритетная операция или скобка или стек операций не опустеет. Всё это работает и под конец он не заносит последнее число в стек, так как занос происходит только тогда, когда char c равна знаку, а после последнего числа знака нет, поэтому я к задаче в конце прибавляю 0, это нам погоды особо не делает, зато все нужные действия выполняются в форе. На всякий случай я закидываю последнее число в стек. И потом наконец используя while выполняю все оставшиеся операции, пока стек операций не опустеет, в таком случае в стеке чисел останется одно число – ответ.

Работа с –file. Пользователь при запуске программы вводит помимо флага, ещё и путь к файлу без русских символов , а так для счёта используется функция –forward/

Работа с –reverse. Я создаю только стек numb для чисел. Далее я получаю на вход пример и начинаю его последовательно считывать. Считывание происходит как и в –forward за исключением того, что я не избавляюсь от пробелов. Таким образом всё и считывается. При каждой встрече пробела я записываю число в стек чисел, если есть что записывать (если там минус, то идёт отдельная проверка). При встрече знака действия, я просто его выполняю с двумя последними элементами стека. В итоге я получаю на верху стека ответ.